UDACITY

‹ Return to "Computer Vision Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Landmark Detection & Tracking (SLAM)

REVIEW

CODE REVIEW 1

HISTORY

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

Hello Udacious Learner, 😄🤗
Saying that I enjoyed going through your work is an understatement. It is almost unbelievable that this is your first submission and you did so well. Your code is clean, modular and you have indeed understood the fundamentals of landmark detection and tracking.
However, there is a slight change you have to make and your project will be completed. Please checkout the suggestions I made and I am confident you will be able to do the required corrections. I also hope the explanations I gave are clear enough.
Continue in this spirit and I wish you all the best as in your next submission

`robot_class.py`: Implementation of `sense`

Implement the `sense` function to complete the robot class found in the `robot_class.py` file. This implementation should account for a given amount of `measurement_noise` and the `measurement_range` of the robot. This function should return a list of values that reflect the measured distance (dx, dy) between the robot's position and any landmarks it sees. One item in the returned list has the format: `[landmark_index, dx, dy]`.

Great attempt implementing the sense function. However, some changes have to be made:
From your notebook, your final pose is [44.725805820112384, 163.37454486121362] and the original pose is [12.08552,18.45997], which is quite far apart different.

- There is a logical error in your if statement. You used an `or` instead of `and`.

```
if (dx <= self.measurement_range and dy <=self.measurement_range):
```

- Also, we wouldn't want negative distances. So it is required that after calculating the distances, you get the absolute values.

```
if (abs(dx )<= self.measurement_range and abs(dy )<=self.measurement_ran
ge):
```

- Finally, we want all the landmarks to be visible at all times, hence we should consider **setting the measurement_range to -1** and therefore we can modify our if statement to something like:

```
if self.measurement_range == -1 or abs(dx )<= self.measurement_range and
abs(dy )<=self.measurement_range:
```

Notebook 3: Implementation of `initialize_constraints`

Initialize the array `omega` and vector `xi` such that any unknown values are `0` the size of these should vary with the given `world_size`, `num_landmarks`, and time step, `N`, parameters.

Great job initializing the constraints as it has been calculated according to the constraint matrix of 2 * (+ num_landmarks) dimension.

Notebook 3: Implementation of `slam`

The values in the constraint matrices should be affected by sensor measurements *and* these updates should account for uncertainty in sensing.

Sensor measurements affect the values in the constraint matrices and the updates do account for uncertainty in sensing

The values in the constraint matrices should be affected by motion `(dx, dy)` *and* these updates should account for uncertainty in motion.

Kudos. The constraint matrices are indeed affected by motion `(dx,dy)` and just like the above, the updates account for uncertainty in motion.

The values in `mu` will be the x, y positions of the robot over time and the estimated locations of landmarks in the world. `mu` is calculated with the constraint matrices `omega^(-1)*xi`.

Nice work. `mu` is calculated with the constraint matrices `omega^(-1)*xi`

Compare the `slam`-estimated and *true* final pose of the robot; answer why these values might be different.

Good job comparing the true pose and the estimated pose. But these values are too far apart. Please do the corrections that have been pointed out in the code review section and the difference between these values would not be very significant

There are two provided test_data cases, test your implementation of slam on them and see if the result matches.

All two test cases pass. Just Brilliant
There are small differences that may have to do with floating point or matrix inverse calculations.

RESUBMIT

DOWNLOAD PROJECT

1 CODE REVIEW COMMENTS ›

Learn the best practices for revising and resubmitting your project.

Rate this review